5Ш

УДК 004.051

В.И. Шинкаренко
Днепропетровский национальный университет железнодорожного транспорта
им. акад. В. Лазаряна, г. Днепропетровск, Украина

Знание-ориентированный подход к адаптации алгоритмов

Предлагается методика формирования альтернативно адаптивных алгоритмов. Знания вырабатываются с использованием показателей степени превосходства и области превосходства алгоритмов. Для адаптации к исходным данным алгоритмов выявляются их атрибуты, по ним исходные данные кластеризуются методом максиминного расстояния. Разработано программное обеспечение для построения альтернативно адаптивных алгоритмов. Экспериментально исследованы возможности их применения к решению задачи сжатия данных без потерь.

Введение

Для решения многих задач наука и практика выработали целый ряд методов решения, иногда значительно отличающихся друг от друга применяемыми подходами, математическим аппаратом и способами обоснования.

Алгоритмы, реализующие различные методы решения одной задачи, образуют некоторое множество функционально эквивалентных (или функционально близких) алгоритмов.

Предметом данного исследования являются функционально эквивалентные, нечетко специфицированные [1] алгоритмы, такие, как алгоритмы распознавания образов, кластеризации, сжатия данных и др. К таким алгоритмам можно отнести практически все алгоритмы искусственного интеллекта. Они отличаются от четко специфицированных тем, что могут выполнять свое функциональное назначение не однозначно, а лишь в некоторой мере хорошо.

Функциональная эффективность алгоритмов зависит от множества входных данных, к которому они применяются.

Предлагается, с целью повышения функциональной эффективности алгоритмов без дополнительных затрат по их совершенствованию, применить к решению задач адаптивные алгоритмы, в терминах [2] – алгоритмы альтернативной адаптации.

Такая адаптация позволяет взять лучшее, что есть в каждом алгоритме и во всех конкретных случаях применять наиболее подходящий алгоритм.

Постановка задачи

Пусть имеется несколько (N) нечетко специфицированных алгоритмов [1] с одинаковым функциональным назначением. Для определенности и конкретности далее будем рассматривать достаточно представительный класс алгоритмов сжатия данных без потерь. Как известно, сжатие данных принципиально невозможно, так как это практически сводится к задаче преобразования вектора из одного пространства в другое, меньшей размерности без потери данных [3]. Сжатие данных с последующим восстановлением возможно лишь при наличии в данных излишней (шумовой, ненужной) информации либо с учетом закономерностей в ее представлении.

Таким образом, любой алгоритм сжатия на конкретных данных может иметь различную функциональную эффективность, проявляющуюся в различной степени сжатия данных.

Область исходных данных, для которых некоторый алгоритм сжатия будет иметь некоторую эффективность не ниже заданной, наперед неизвестно и практически может быть определена лишь экспериментально, т.е. после выполнения алгоритма.

Повысить функциональную эффективность некоторого класса алгоритмов с одинаковым функциональным назначением предлагается путем их адаптации к исходным данным или частям исходных данных.

Необходимо выработать методику построения алгоритма альтернативной адаптации и апробировать ее на алгоритмах сжатия данных без потерь.

Алгоритмы альтернативной адаптации

Обозначим алгоритмы одинаковой функциональной направленности $A_i|_X^Y$, где X – область определения, а Y – область значения алгоритмов. В нашем случае A_i – несколько алгоритмов сжатия, реализованных в программах-архиваторах файлов, $x \in X$ – множество файлов, подлежащих архивации, $y \in Y$ – архив файлов.

Задана $\rho(y) \in R^+$ – оценка качества полученных алгоритмом результатов ($y \in Y$), для алгоритмов сжатия – размер архива.

Идея альтернативной адаптации алгоритмов A_i отражена в следующем адаптивном алгоритме:

$$A_a = A_o|_X^{X_1,X_2\ldots X_N} \cdot A_1|_{X_1}^{Y_1} \cdot A_2|_{X_2}^{Y_2} \cdot \ldots \cdot A_N|_{X_N}^{Y_N} = A_o|_X^{X_1,X_2\ldots X_N} \cdot \prod_{i=1}^{N} A_i|_{X_i}^{Y_i}, \quad (1)$$

где «·» – операция композиции, которая определяет последовательное выполнение алгоритмов (краткая форма записи $\prod_{i=1}^{N} A_i|_{X_i}^{Y_i}$). Алгоритм $A_o|_X^{X_1,X_2\ldots X_N}$ разбивает область определения адаптируемого алгоритма X на подмножества $X_1, X_2 \ldots X_N$, такие, что

$$\bigcup_{i=1}^{N} X_i = X, \forall i \neq j \,|\, X_i \cap X_j = \varnothing \text{ и } \forall y_i \in Y_i, y_j \in Y_j \,|\, \rho(y_i) < \rho(y_j). \quad (2)$$

Алгоритм A_a состоит из адаптирующей части $A_o|_X^{X_1,X_2\ldots X_N}$ и адаптируемой $\prod_{i=1}^{N} A_i|_{X_i}^{Y_i}$. Функциональное назначение адаптируемой части остается тем же – сжатие данных, а адаптирующей – максимально повысить эффективность этой функциональности применительно к конкретным входным данным.

Оптимальным будет такое разбиение множества X, при котором $S(A_a, A_i)|_X$ будет максимальным.

Здесь $S(A_i, A_j)|_X$ – степень превосходства одного (i-го) алгоритма над другим (j-ым) на ограниченном множестве X [1]:

$$S(A_i, A_j)|_X = \frac{1}{N_{x_i \in X}} \sum_{x_i \in X} \frac{\rho(A_j|_{x_i}) - \rho(A_i|_{x_i})}{\max(\rho(A_j|_{x_i}), \rho(A_i|_{x_i}))} \cdot 100\,\%. \quad (3)$$

Задача заключается в разработке алгоритма $A_o|_X^{X_1,X_2\ldots X_N}$, чтобы выполнялись условия (2) и целевая функция (3) была максимальной, с учетом того, что:

$$\rho(A_{X_1 \cup X_2}) \neq \rho(A_{X_1}) + \rho(A_{X_2}). \quad (4)$$

Такой алгоритм существенно зависит от порядка сбора информации, необходимой для адаптации и применения адаптивного алгоритма, что проявляется в режиме адаптации.

Режимы адаптации

Можно говорить о двух режимах адаптации. Назовем СНС (сегодня на сегодня) адаптацией такую, при которой данные, к которым необходимо адаптировать алгоритм, используются адаптирующим алгоритмом.

Если адаптирующий алгоритм использует ранее полученные статистические данные результатов применения адаптируемых алгоритмов, будем говорить о СНЗ (сегодня на завтра) адаптации.

СНЗ адаптация адаптирует алгоритм к последовательности входных данных, при многократном выполнении алгоритма. Адаптация к данным при конкретном выполнении использует знания о результатах предыдущих выполнений алгоритма.

Таким образом, СНС алгоритм адаптируется к самим исходным данным, а СНЗ – к потоку входных данных при последовательности выполнений.

При СНС адаптации адаптирующий алгоритм:

$$A_o = \prod_j \prod_i (A_g |_X^{X_j} \cdot A_i |_{X_j}^{Y_{i,j}} \cdot A_f |_{X_i, Y_{i,j}}^{F_{i,j}}) \cdot A_r |_{X,F}^{X_1, X_2 \dots X_N}, \tag{5}$$

где $A_g |_X^{X_j}$ – алгоритм выделения некоторого подмножества $X_j \subset X$, $A_f |_{X_i, Y_{i,j}}^{F_{i,j}}$ – алгоритм протоколирования результатов сжатия подмножества X_j алгоритмом A_i, $A_r |_{X,F}^{X_1, X_2 \dots X_N}$ – алгоритм разбиения множества X на подмножества $X_1, X_2 \dots X_N$.

Алгоритм A_f предназначен для формирования базы фактов (базы данных). Применительно к алгоритмам сжатия база фактов может состоять из таких кортежей: идентификатор архиватора, атрибуты файла, начальный и конечный размер файла.

Алгоритм определения всех возможных подмножеств некоторого множества принадлежит к классу NP – алгоритмов [4]. Его ($A_g |_X^{X_j}$) вычислительная сложность является $O(n!)$. Такие задачи даже при незначительных размерностях требуют недостижимых вычислительных ресурсов.

Общепринятый прием снижения вычислительной сложности заключается в переходе к приближенным алгоритмам, чаще всего стохастическим, либо упрощении задачи, вводя некоторые допущения.

Пойдем вторым путем. Допустим неравенство (3) можно заменить равенством без большого ущерба для результата. Очевидно, что такое предположение для алгоритмов будет тем более обоснованным, чем больше и однороднее сжимаемые файлы.

Тогда алгоритмом $A_g |_X^{x_i}$ можно выполнить простой перебор элементов $x_i \in X$, т.е. последовательный выбор файлов для сжатия из обучающей выборки.

Алгоритм A_f также упрощается. Элемент x_j добавляется к множеству X_i так, что $x \in X_i \mid \rho(A_i |_x) = \max_j \rho(A_j |_x)$.

Адаптирующий алгоритм СНЗ адаптации состоит из двух частей:

$$A_{o,1} = \prod_{j=1}^{M} \prod_{i=1}^{K \le N} (A_g |_X^{X_j} \cdot A_i |_{X_j}^{Y_{i,j}} \cdot A_f |_{X_i, Y_{i,j}}^{F_{i,j}}) \cdot A_z |_F^Q \text{ и } A_{o,2} = A_r |_{X,Q}^{X_1, X_2 \dots X_n}, \tag{6}$$

где A_z – алгоритм извлечения знаний из базы фактов F и формирования базы знаний Q.

Алгоритм $A_{o,1}$ может выполняться заблаговременно, до использования адаптивного алгоритма, однократно или периодически, а $A_{o,2}$ – как часть адаптивного алгоритма при необходимости сжатия данных.

Конечно, важным вопросом является вопрос эффективности альтернативной адаптации. Для ответа на него выполнен ряд вычислительных экспериментов.

Экспериментальные исследования

Разработан исследовательский комплекс программ ResComp, позволяющий выполнять численные эксперименты с отбором файлов для сжатия, выполнения самого сжатия, определения размеров исходных файлов и архивов.

Для формирования адаптивных алгоритмов сжатия данных отобрано несколько алгоритмов и их модификаций, реализованных в 12 архиваторах данных, имеющих возможность запуска с командной строки (для выполнения из программы ResComp).

Каждый архиватор имеет от нескольких до десятков параметров, влияющих на содержимое архивов и степень их сжатия. В данной работе параметры архиваторов фиксированы и выбраны согласно рекомендациям в их help‘е, которые предполагают наилучшую степень сжатия. Ниже перечислены архиваторы с указанием разработчиков, версии, года выпуска и параметров командной строки для запуска архиваторов:

- Rar 3.71, Alexander Roshal, 2007, “a -m5 bbb all”;
- Ha 0999c, Harry Hirvola, 1995, “a2 bbb all*.*”;
- Arj 3.14a, ARJ Software, 2006, “a -jm '+ ' bbb all”;
- 7_zip 4.58 beta, Igor Pavlov, 2008, “a -t7z bbb all -mx9”;
- Bzip2 1.0.4, Julian Seward, 2006, “--best -k all*.*”;
- AlZip 7.0 beta1, ESTSoft corp, 2007, “-a -m0 all bbb.alz”;
- Rk 1.04.1 alpha, Malcolm Taylor, 2000, “-c bbb.rk @ccc2.txt”;
- Tar 1.12, “-c -k -z --files-from=ccc1.txt --file=bbb.tar”;
- Imp 1.12, Technelysium Pty ltd., 2000, “a -m3 bbb.imp all*.*”;
- Jar 1.02, ARJ Software, 1997, “a -m4 bbb.jar all*.*”;
- PKZIPC(R)_4.00, PKWARE inc., 2000, “-add bbb.zip all*.*”;
- Gzip 1.3.12, Jean-loup Gailly, “--best all*.*”.

Здесь “bbb” – имя архива, “ccc” – имя файла со списком файлов, подлежащих архивации, “all” – имя папки с исходными файлами.

Сформировано 7 банков данных файлов различных форматов [5] (общая информация в табл.1).

Таблица 1 – Основные характеристики банков данных

Банк данных	Формат файлов (расширение файлов)	К-во файлов	Объем БД, Гб	Минимальный размер файла, б	Максимальный размер файла, Мб
1	BitMap File (bmp)	3037	2,03	104	2,25
2	“Deja-vu” (djvu)	513	1,19	8990	21,6
3	Microsoft Word(doc)	3611	1,31	0	23,0
4	Zsoft PaintBrush File (pcx)	1016	1,73	291	5,23
5	Portable Document Format (pdf)	566	0,79	5955	28,8
6	Tag Image File (tif)	1474	1,42	188	20,2
7	Microsoft Excel(xls)	6523	1,86	136	20,7
Всего		16740	10,33	0	28,8

Выполнено два численных эксперимента.

Подготовка к экспериментам заключается в формировании таблицы фактов. На этом этапе выполнен алгоритм $A_{o,1}$ применительно ко всем файлам банка данных и всех архиваторов.

Алгоритм $A_g |_X^{X_j}$ (5 – 6) реализует последовательную выборку одиночного файла из банка данных.

Файлы x_i обладают некоторыми свойствами – атрибутами, (x_i, a_i) – входные данные для однократного выполнения алгоритма и a_i – набор атрибутов $(x_i, a_i) \in X \times A$.

Для всех данных набор атрибутов включает формат файла и, в зависимости от формата, определяются остальные атрибуты. Так, для файлов формата MS Word, определяется количество символов, количество строк, количество внедренных объектов MS Excel, количество внедренных графических объектов.

Алгоритм $A_f\,|^{F_{i,j}}_{X_i,Y_{i,j}}$ (5 – 6) определяет набор и значения атрибутов файла и формирует базу фактов со следующим содержимым: имя файла, его формат, атрибуты файла, размер архива при сжатии каждым архиватором, лучший архиватор для данного файла.

Эксперимент 1. Исследуется СНС и СНЗ адаптация со знаниями на уровне форматов файлов.

Для СНЗ адаптации алгоритм $A_z\,|^{Q}_{F}$ определяет степень превосходства (3) и область превосходства [1] алгоритмов архивации отдельно для файлов различного формата. Значения степени превосходства $S(A_i, A_j)|_X$ для файлов формата MS Word с довериительными интервалами [1], [6], приведены в табл. 2. Есть явный «лидер» – алгоритм A_4.

Алгоритм $A_z\,|^{Q}_{F}$ формирует базу знаний следующего формата:

если «формат данных = *a*», то лучший архиватор *b*. Например, если «формат данных = *DO*», то лучший архиватор A_4.

Результаты эксперимента показали, что для каждой структуры данных выявлен алгоритм, превосходящий остальные на 10 … 20 %.

Однако по области превосходства проявились различия. Для файлов форматов xls, doc, pdf, djvu область превосходства лидера над остальными алгоритмами находится в пределах 86 … 99,8 %. То есть лидер был лучшим практически всегда, при архивации подавляющего большинства файлов. Иначе лидер появился при архивации файлов форматов tif, pcx и bmp. Его превосходство над некоторыми другими архиваторами составила лишь порядка 43 … 52 %. Здесь лидер был хуже практически для половины файлов, хотя по степени превосходства может и явно превосходил другие. В табл. 3 показано насколько часто каждый архиватор был лучшим при сжатии данных определенного формата.

Таблица 2 – Степень превосходства i-го алгоритма над j-м

i \ j	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
A_1	0,0	$16,6^{+3,0}_{-3,0}$	$15,0^{+1,9}_{-1,9}$	$-7,9^{+2,9}_{-2,9}$	$11,6^{+1,2}_{-1,2}$	$15,3^{+3,3}_{-3,3}$	$9,0^{+4,3}_{-4,3}$	$15,0^{+2,6}_{-2,6}$	$6,0^{+2,7}_{-2,7}$	$9,8^{+2,2}_{-2,2}$	$14,4^{+2,2}_{-2,2}$	$12,9^{+3,2}_{-3,2}$
A_2	$-16,6^{+3,0}_{-3,0}$	0,0	$-2,6^{+1,9}_{-1,9}$	$-22,9^{+0,9}_{-0,9}$	$-6,3^{+3,7}_{-3,7}$	$-2,1^{+1,1}_{-1,1}$	$-8,7^{+1,7}_{-1,7}$	$-2,6^{+1,2}_{-1,2}$	$-11,4^{+1,7}_{-1,7}$	$-7,9^{+4,5}_{-4,5}$	$-3,2^{+1,6}_{-1,6}$	$-4,7^{+1,1}_{-1,1}$
A_3	$-15,0^{+1,9}_{-1,9}$	$2,6^{+1,9}_{-1,9}$	0,0	$-21,6^{+0,9}_{-0,9}$	$-4,0^{+2,5}_{-2,5}$	$0,5^{+2,0}_{-2,0}$	$-6,6^{+2,7}_{-2,7}$	$-3,4^{+1,0}_{-1,0}$	$-9,6^{+0,6}_{-0,6}$	$-5,6^{+3,1}_{-3,1}$	$-0,7^{+0,4}_{-0,4}$	$-2,3^{+1,7}_{-1,7}$
A_4	$7,9^{+2,9}_{-2,9}$	$22,9^{+0,9}_{-0,9}$	$21,6^{+0,9}_{-0,9}$	0,0	$18,5^{+3,3}_{-3,3}$	$21,9^{+0,8}_{-0,8}$	$16,2^{+1,3}_{-1,3}$	$21,6^{+0,3}_{-0,3}$	$13,5^{+0,9}_{-0,9}$	$17,1^{+3,8}_{-3,8}$	$21,2^{+0,8}_{-0,8}$	$19,7^{+0,6}_{-0,6}$
A_5	$-11,6^{+1,2}_{-1,2}$	$6,3^{+3,7}_{-3,7}$	$4,0^{+2,5}_{-2,5}$	$-18,5^{+3,3}_{-3,3}$	0,0	$4,5^{+4,2}_{-4,2}$	$-2,7^{+5,3}_{-5,3}$	$4,0^{+3,4}_{-3,4}$	$-5,9^{+3,4}_{-3,4}$	$-1,7^{+1,3}_{-1,3}$	$3,3^{+2,9}_{-2,9}$	$1,9^{+4,0}_{-4,0}$
A_6	$-15,3^{+3,3}_{-3,3}$	$2,1^{+1,1}_{-1,1}$	$-0,5^{+2,0}_{-2,0}$	$-21,9^{+0,8}_{-0,8}$	$-4,5^{+4,2}_{-4,2}$	0,0	$-7,1^{+0,8}_{-0,8}$	$-0,5^{+1,0}_{-1,0}$	$-10,0^{+1,3}_{-1,3}$	$-6,2^{+4,9}_{-4,9}$	$-1,2^{+1,6}_{-1,6}$	$-2,8^{+0,3}_{-0,3}$
A_7	$-9,0^{+4,2}_{-4,2}$	$8,7^{+1,7}_{-1,7}$	$6,6^{+2,7}_{-2,7}$	$-16,2^{+1,3}_{-1,3}$	$2,7^{+5,3}_{-5,3}$	$7,1^{+0,8}_{-0,8}$	0,0	$6,6^{+1,7}_{-1,7}$	$-3,2^{+2,0}_{-2,0}$	$1,0^{+5,7}_{-5,7}$	$6,0^{+2,3}_{-2,3}$	$4,4^{+1,0}_{-1,0}$
A_8	$-15,0^{+2,6}_{-2,6}$	$2,6^{+1,2}_{-1,2}$	$3,4^{+1,0}_{-1,0}$	$-21,6^{+0,3}_{-0,3}$	$-4,0^{+3,4}_{-3,4}$	$0,5^{+1,0}_{-1,0}$	$-6,6^{+1,7}_{-1,7}$	0,0	$-9,6^{+0,4}_{-0,4}$	$-5,6^{+4,0}_{-4,0}$	$-0,7^{+0,7}_{-0,7}$	$-2,2^{+0,7}_{-0,7}$
A_9	$-6,0^{+2,7}_{-2,7}$	$11,4^{+1,7}_{-1,7}$	$9,6^{+0,6}_{-0,6}$	$-13,5^{+0,9}_{-0,9}$	$5,9^{+3,4}_{-3,4}$	$10,0^{+1,3}_{-1,3}$	$3,2^{+2,0}_{-2,0}$	$9,6^{+0,4}_{-0,4}$	0,0	$4,1^{+3,7}_{-3,7}$	$9,0^{+0,3}_{-0,3}$	$7,4^{+1,1}_{-1,1}$
A_{10}	$-9,8^{+2,2}_{-2,2}$	$7,9^{+4,5}_{-4,5}$	$5,6^{+3,1}_{-3,1}$	$-17,1^{+3,8}_{-3,8}$	$1,7^{+1,3}_{-1,3}$	$6,2^{+4,9}_{-4,9}$	$-1,0^{+5,7}_{-5,7}$	$5,6^{+4,0}_{-4,0}$	$-4,1^{+3,7}_{-3,7}$	0,0	$5,0^{+3,5}_{-3,5}$	$3,7^{+4,7}_{-4,7}$
A_{11}	$-14,4^{+2,2}_{-2,2}$	$3,2^{+1,6}_{-1,6}$	$0,7^{+0,4}_{-0,4}$	$-21,2^{+0,8}_{-0,8}$	$-3,3^{+2,9}_{-2,9}$	$1,2^{+1,6}_{-1,6}$	$-6,0^{+2,3}_{-2,3}$	$0,7^{+0,7}_{-0,7}$	$-9,0^{+0,3}_{-0,3}$	$-5,0^{+3,5}_{-3,5}$	0,0	$-1,5^{+1,3}_{-1,3}$
A_{12}	$-12,9^{+3,2}_{-3,2}$	$4,7^{+1,1}_{-1,1}$	$2,3^{+1,7}_{-1,7}$	$-19,7^{+0,6}_{-0,6}$	$-1,9^{+4,0}_{-4,0}$	$2,8^{+0,3}_{-0,3}$	$-4,4^{+1,0}_{-1,0}$	$2,2^{+0,7}_{-0,7}$	$-7,4^{+1,1}_{-1,1}$	$-3,7^{+4,7}_{-4,7}$	$1,5^{+1,3}_{-1,3}$	0,0

Таблица 3 – Распределение лучших архиваторов по форматам файлов

Архиватор	Часть файлов, для которых архиватор оказался лучшим, %						
	bmp	djvu	doc	pcx	pdf	tif	xls
A_1	35,3	2,7	7,0	4,8	1,8	28,4	4,8
A_2	35,2	2,5	1,7	29,9	0	34,4	1,5
A_4	14,9	0	90,4	6,3	83,4	22,3	93,3
A_5	6,3	0	0,4	32,7	0,2	9,6	0,4
A_7	4,0	0	0	26,4	1,6	0,4	0
A_9	0	87,7	0	0	12,8	0,2	0
A_{12}	4,4	7,0	0	0	0,4	4,6	0

В ходе эксперимента выполнено порядка 4000 опытов. Каждый опыт состоял в том, что случайным образом из банка файлов отбиралось от 1 до 500 файлов. При этом две трети банка использовались в качестве обучающей выборки, одна треть – контрольной. Последовательно выполнялось сжатие каждым архиватором, СНС и СНЗ адаптивными архиваторами с пополнением базы фактов. СНС адаптивный архиватор выполнял сжатие согласно базы фактов предварительно определенным лучшим архиватором. СНЗ адаптивный архиватор сжимал файлы согласно их форматов и знаниям о лучшем архиваторе для данного формата.

Результаты эксперимента показали, что СНЗ адаптивный алгоритм в любых случаях и вариантах выполнения имеет положительную степень превосходства над отдельными архиваторами. При этом в для смеси файлов различной структуры это превосходство достаточно небольшое, порядка 5 … 10 %. При гомонизации смеси, в случае неудачно выбранного архиватора, степень превосходства СНЗ адаптивного алгоритма может доходить до 20 % и более, что наглядно представлено в табл. 4.

Таблица 4 – Степень превосходства СНЗ адаптивного архиватора

Архиватор	Форматы файлов							
	bmp	djvu	doc	pcx	pdf	tif	xls	смесь файлов
A_1	0	0,2	8	14	1	0	14	5
A_2	7	0,8	23	0	5	3	21	8
A_4	11	1,1	0	4	0	5	0	4
A_9	23	0	14	16	1	16	20	10
Лучший из A_1 … A_{12}	0	0	0	0	0	0	0	3

Интересно, что степень превосходства СНС адаптивного алгоритма по отношению СНЗ адаптивного составляет всего 0,5 %. Учитывая то, что реализация адаптирующего алгоритма в СНС адаптивном требует значительных вычислительных затрат, и то, что эти затраты необходимы в момент потребности в архивации этот факт является существенным. При СНЗ адаптации сбор фактов и построение базы знаний можно выполнить заранее. При этом время архивации СНЗ адаптивным архиватором практически такое же, как обычным, а результат всегда лучше и часто значительно лучше.

Эксперимент 2. Выполняется адаптация алгоритмов к содержимому файлов.

Знания, необходимые для адаптации, основаны на значениях атрибутов файлов и предварительно полученных результатах сжатия. Так как атрибуты у различных форматов файлов различны, то естественно строить адаптивный алгоритм с учетом форматов.

Не целесообразно применять адаптивный алгоритм для файлов формата doc, xls, djvu, pdf, что видно из предыдущего эксперимента. Для этих форматов выявлен архиватор, явный лидер, который превосходит остальные почти всегда, и замена его другим в тех редких случаях, когда он уступает, очень мало влияет на усредненный показатель степени превосходства.

Остальные исследуемые форматы относятся к графическим. Рассмотрим адаптацию архиваторов к файлам формата tif. Он имеет большой набор внешних атрибутов и в этом случае три архиватора в основном делят множество файлов на части, в которых они лучшие.

Для исследования выбраны следующие атрибуты: размер файла, высота и ширина изображения в пикселах, модель цвета, глубина цвета, способ сжатия, вертикальное и горизонтальное разрешение.

При подготовке к экспериментам значения атрибутов вместе с результатами сжатия занесены в базу фактов.

База знаний формировалась следующим образом. Значения атрибутов нормализовались. Выполнялась кластеризация методом максиминного расстояния [7], с особенностью в том, что новые кластеры образуются, если расстояние до нового центра кластера превышает 1/6 среднего межкластерного расстояния (Евклидова).

Анализ показал, что порядка 70 … 80 % кластеров имеют своего лидера среди архиваторов, который более чем в 90 % файлов кластера превосходит остальные.

Эксперимент заключался в отборе 40 … 150 файлов в качестве обучающей выборки и 1 … 150 – в качестве контрольной.

Как оказалось, степень превосходства СНЗ адаптивного алгоритма над алгоритмом A_1 (лучшим из остальных) составляет 1 … 2 %.

Хотя полученная система знаний хорошо отражает объект моделирования, низкий результат объясняется тем, что область превосходства алгоритмов A_1 … A_{12} приходится на файлы небольших размеров и результаты их сжатия на средние размеры влияют мало. Кроме того, неравенство (4) проявилось достаточно сильно.

Выводы

Предложенная методика формирования альтернативно адаптивного алгоритма позволяет повысить функциональную эффективность множества функционально эквивалентных алгоритмов. Применение СНЗ адаптивных алгоритмов требует предварительной подготовки и практически не влияет на временную эффективность алгоритмов, что очень важно ввиду больших потребностей в вычислениях алгоритмов искусственного интеллекта, для которых, в основном, и предназначена данная работа.

Существенным преимуществом СНЗ адаптации является:

- относительно высокая временная эффективность адаптирующей части алгоритма;
- эксплуатация адаптивного алгоритма в режиме близким к алгоритмам без адаптации;
- способность алгоритма к дообучению и повышению усредненных показателей функциональной эффективности при длительной эксплуатации.

Предложен критерий оценки эффективности применения знаний в альтернативно адаптивных алгоритмах с применением показателей степени и области превосходства алгоритмов. Эти показатели позволяют оценить потенциальные возможности применения альтернативной адаптации алгоритмов.

Проявление лидера среди альтернативных алгоритмов, имеющего существенную степень превосходства над всеми другими алгоритмами и область превосходства более 90 %, говорит о нецелесообразности адаптации в данной области применения алгоритмов.

Ускорить обучение и повысить эффективность адаптирующего алгоритма можно за счет знаний, полученных из нескольких источников при применении таких алгоритмов в различных программных продуктах и централизованном сборе информации в единой базе фактов.

Направления дальнейших исследований представляются в следующем.

Естественно, при первом выполнении алгоритма A_g база знаний пуста. В этом случае и в случае, когда знаний недостаточно, необходимо предусмотреть некоторую стратегию выбора алгоритма сжатия. В качестве такой стратегии может быть предусмотрено поочередное выполнение алгоритмов, случайный выбор алгоритма, выбор в соответствии с наперед заданными (и изменяемыми в последствии) приоритетами.

Множество различных решений имеет и задача формирования базы знаний.

Отметим, что алгоритмы A_g и A_z также могут адаптироваться к своим данным. В этом случае будем иметь двухуровневую адаптацию. На первом уровне адаптируется алгоритм заданного функционального назначения, а на втором, одновременно с ним – адаптирующий алгоритм.

Литература

1. Шинкаренко В.И. Функциональная эффективность нечетко специфицированных алгоритмов // Проблемы программирования. – 2006. – № 1. – С. 24-33.
2. Растригин Л.А. Адаптация сложных систем. – Рига: Зинатне, 1981. – 375 с.
3. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: Диалог-МИФИ, 2002. – 384 с.
4. Успенский В.А., Семенов А.Л. Теория алгортмов: основные открытия и приложения. М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 288 с.
5. Борн Г. Форматы данных. – К.: Торгово-издательское бюро BHV, 1995. – 472 с.
6. Соболь И.М. Численные методы Монте-Карло. – М.: Наука. Гл. ред. физ.-мат. лит., 1973. – 311 с.
7. Ту Дж., Гонсалес Р. Принципы распознавания образов. – 1978. – 411 с.

В.І. Шинкаренко

Підхід до адаптації алгоритмів, орієнтований на знання

Пропонується методика формування альтернативно адаптивних алгоритмів. Знання визначаються з застосуванням показників ступеня та області переваги алгоритмів. Для адаптації до вхідних даних алгоритмів виявляються їх атрибути, за ними вхідні дані кластеризуються методом максимінних відстаней. Розроблене програмне забезпечення для формування альтернативно адаптивних алгоритмів. Експериментально досліджені можливості їх застосування при стисканні файлів без втрат.

V.I. Shynkarenko

Knowledge-Oriented Approach to Adaptation of Algorithms

The method to alternatively adaptive algorithm‘s designing is offered. Knowledge extracts with use such measures as degree of the superiority and area of the superiority of algorithms. The attributes of source data of algorithm‘s are detected and clustered by method of minimum and maximum distances. The specific software is developed for designing alternatively adaptive algorithms. Facilities of their application to the decision of a problem of lost-free compression of data are experimentally investigated.

Статья поступила в редакцию 18.07.2008.